

# FiEstAS sampling – a Monte Carlo algorithm for multidimensional numerical integration

Yago Ascasibar

*Astrophysikalisches Institut Potsdam  
An der Sternwarte 16, Potsdam D-14482, Germany*

*Universidad Autónoma de Madrid  
Dpto Física Teórica, Campus de Cantoblanco, Madrid E-28049, Spain*

---

## Abstract

This paper describes a new algorithm for Monte Carlo integration, based on the Field Estimator for Arbitrary Spaces (FIESTAS). The algorithm is discussed in detail, and its performance is evaluated in the context of Bayesian analysis, with emphasis on multimodal distributions with strong parameter degeneracies. Source code is available upon request.

*Key words:* numerical integration, Bayesian inference, Monte Carlo importance sampling

*PACS:* 02.60.Jh, 02.50.Tt, 02.70.Uu

---

## 1 Introduction

Many problems in Physics, as well as in other branches of science, involve the computation of an integral

$$I = \int_V f(\mathbf{x}) \, d\mathbf{x} \quad (1)$$

over a given multidimensional domain  $V$ . In most circumstances, the value of  $I$  has to be estimated numerically, evaluating the function  $f$  at a series of  $N$  sample points  $\mathbf{x}_i$ .

Numerical quadrature algorithms (see e.g. [1]) sample  $f$  on a regular grid or at a carefully selected set of unequally-spaced points (Gaussian quadrature).

---

*Email address:* `yago.ascasibar@uam.es` (Yago Ascasibar).

This approach is very efficient for smooth functions, but it is certainly far from optimal when the integrand is strongly peaked. Adaptive techniques provide a solution to this problem by refining those sub-intervals of the integration domain where  $f$  is found to vary on smaller scales.

In multiple dimensions, Monte Carlo methods usually yield better accuracy for a given  $N$  than repeated integrations using one-dimensional deterministic algorithms. The essence of the Monte Carlo method is that the sample points  $\mathbf{x}_i$  are chosen (to some extent) at random. The most basic version uses a uniform probability distribution over the integration region, and the estimator for the integral  $I$  is given by

$$\hat{I}_{\text{MC}} = \frac{V}{N} \sum_{i=1}^N f_i \quad (2)$$

where  $V$  denotes the integration volume and  $f_i \equiv f(\mathbf{x}_i)$ . The error associated to  $\hat{I}_{\text{MC}}$  can be estimated as

$$\hat{\Delta}_{\text{MC}}^2 = \frac{V^2}{N^2} \sum_{i=1}^N \left( f_i - \frac{\hat{I}_{\text{MC}}}{V} \right)^2. \quad (3)$$

Although the error decreases as  $N^{-1/2}$  (this is, in fact, why Monte Carlo methods are preferable in multidimensional problems), the basic Monte Carlo is still computationally inefficient when the function  $f$  has a very narrow peak compared to the total integration volume. On the one hand, a large number of sample points would be required to obtain good estimates  $\hat{I}_{\text{MC}}$  and  $\hat{\Delta}_{\text{MC}}$ . On the other hand, and more important, it is indeed quite possible that the algorithm achieves convergence, reporting a wrong value of the integral with an extremely small estimate of the error, before finding the maximum (or maxima) of  $f$ .

A refinement of this method, known as importance sampling, consists in drawing the points from a distribution similar in form to the integrand, so that they are more likely to come from the regions that contribute most to  $I$ . Rather than sampling the whole integration volume uniformly, one chooses a probability distribution  $g(\mathbf{x})$  that adapts to the shape of the integrand. For any  $g(\mathbf{x})$ , the integral (1) can be re-written as

$$I = \int_V \frac{f(\mathbf{x})}{g(\mathbf{x})} g(\mathbf{x}) \, d\mathbf{x} \quad (4)$$

and the estimators of  $I$  and its error become

$$\hat{I}_{\text{IS}} = \frac{1}{N} \sum_{i=1}^N \frac{f_i}{g_i} \quad (5)$$

and

$$\hat{\Delta}_{\text{IS}}^2 = \frac{1}{N^2} \sum_{i=1}^N \left( \frac{f_i}{g_i} - \hat{I}_{\text{IS}} \right)^2 \quad (6)$$

respectively. The key issue is, of course, the choice of the distribution  $g(\mathbf{x})$ . These two expressions reduce to (2) and (3) for the particular case of uniform sampling, i.e. all  $g_i \equiv g(\mathbf{x}_i) = 1/V$ . In the ideal case, when  $g(\mathbf{x}) = |f(\mathbf{x})|/I$ , all points contribute exactly the same amount to the sum in (5) and zero in (6), and the algorithm takes only a few  $N$  to achieve the desired accuracy. Unfortunately, choosing  $g(\mathbf{x})$  is not straightforward when the shape of  $f$  is unknown *a priori*, and evaluating it is actually part of our goal.

A perfect example of such a situation is Bayesian inference, where one tries to select the model  $M$  that best describes the data  $\mathbf{D}$  and estimate the values of the model parameters  $\Theta$ . For a given model, Bayes' Theorem states that

$$p(\Theta|M, \mathbf{D}) = \frac{p(\Theta|M) p(\mathbf{D}|M, \Theta)}{p(\mathbf{D}|M)} \quad (7)$$

where  $p(\Theta|M)$  is the prior probability distribution of the model parameters,  $p(\mathbf{D}|M, \Theta)$  is the likelihood of the data,  $p(\Theta|M, \mathbf{D})$  is the posterior probability distribution of the parameters, and  $p(\mathbf{D}|M)$  is the evidence for model  $M$ ,

$$p(\mathbf{D}|M) = \int p(\Theta|M) p(\mathbf{D}|M, \Theta) d\Theta \quad (8)$$

The posterior probability distribution (7) contains all the information needed for parameter estimation, and the evidence plays in this case the role of a mere normalization constant. Nevertheless, the value of the integral (8) becomes critical in model selection. Applying again Bayes' Theorem, two models  $M_1$  and  $M_2$  can be compared by evaluating the Bayes' factor,

$$\frac{p(M_1|\mathbf{D})}{p(M_2|\mathbf{D})} = \frac{p(M_1) p(\mathbf{D}|M_1)}{p(M_2) p(\mathbf{D}|M_2)} \quad (9)$$

where  $p(M_i)$  denotes the prior probability of each model,  $p(\mathbf{D}|M_i)$  are their evidences, and  $p(M_i|\mathbf{D})$  are the posterior probabilities. Even if one is only interested in parameter estimation, it is good practice to quote the evidence (as well as the assumed priors) in order to make possible the comparison with future work.

The computational part of a Bayesian analysis consists thus in the evaluation of the integral  $I = p(\mathbf{D}|M) = \int f(\Theta) d\Theta$ , where  $f(\Theta) = p(\Theta|M) p(\mathbf{D}|M, \Theta)$ , over the region of the parameter space where  $p(\Theta|M) > 0$ . Since each evaluation of the likelihood may involve a significant computational cost, the algorithm should locate the the maxima of  $f(\Theta)$  as efficiently as possible, even when this function has a complicated structure. In particular, it is not uncommon that  $f(\Theta)$  presents several maxima, corresponding to different sets

of parameter values that provide a good fit to the data. Which solution is to be preferred depends, within the Bayesian framework, on the value of the integral (the evidence) over the region associated to each peak. Even more often, there are strong degeneracies in parameter space. The maxima of the function  $f$  can be extremely elongated, generally with a curved shape arising from a non-linear relationship between two or more parameters.

These two problems (multimodality and curved degeneracies) have proven difficult to overcome for many standard algorithms, especially as the number of dimensions increases. Markov Chain Monte Carlo methods (see e.g. [2]) typically require a relatively large number of evaluations. Nested sampling [3] provides a more efficient alternative, transforming the multidimensional integral (1) into a single dimension by means of a re-parameterization in terms of the ‘prior mass’  $p(\Theta|M) d\Theta$ . This formalism has recently been extended to cope with the problems of multimodality and curved degeneracies by resorting to clustering algorithms [4,5]. On the other hand, traditional importance sampling methods (e.g. Vegas [6]) assume a separable weight function, i.e.  $g(\mathbf{x}) = \prod_{d=1}^D g_d(x_d)$ , which can give a good approximation to  $f(\mathbf{x})$  only as long as the characteristic features are well aligned with the coordinate axes. An adaptive refinement technique has been implemented in the algorithm Suave [7] to improve the performance in the general case.

Here I present a new method, based on the Field Estimator for Arbitrary Spaces (FIESTAS) developed by [8] to estimate the continuous density field underlying a given point distribution. The algorithm is fully described in Section 2, results are presented in Section 3, and the main conclusions are summarized in Section 4.

## 2 Description of the algorithm

FIESTAS sampling performs multidimensional numerical integration by means of Monte Carlo importance sampling. Here I discuss in detail the most relevant algorithmic issues; actual source code is available upon request.

The method has three free parameters: the desired relative accuracy  $\epsilon$ , the fraction of points that are uniformly distributed  $\eta_u$ , and the fractional increase in the number of points per step  $\eta_n$ . Default values are  $\epsilon = 0.01$ ,  $\eta_u = 0.1$ , and  $\eta_n = 0.1$ . For the first step, the number of new sample points is set to  $n = 1$ .

The main loop of the program can be summarized as

- (1) generate  $\eta_u n$  new points uniformly
- (2) compute  $g(\mathbf{x})$  for this step using FIESTAS

- (3) sample  $n_F = (1 - \eta_u)n$  new points from  $g(\mathbf{x})$
- (4) evaluate the integral and its error
- (5) increase  $n$  by a fraction  $\eta_n$
- (6) repeat until the desired accuracy is achieved

## 2.1 Uniform sampling

FiESTAS sampling is intended to adapt the probability distribution  $g(\mathbf{x})$  to the shape of the integrand within the fewest possible number of iterations. As will be shown below, the presence of pronounced curved degeneracies does not pose a significant problem to the algorithm, but the discovery of narrow, isolated maxima is, at best, a very hard problem, and a large number of sample points may be required in order to achieve the correct result. Even worse, there can be *no* guarantee that *all* the maxima of  $f$  have been located, and therefore it is important to reach a compromise between the detection of potential multimodality and sampling efficiency.

Sampling from a uniform distribution is obviously the most unbiased way to explore the integration domain in search of previously unknown maxima. The fraction of points (i.e. computational effort) devoted to such exploration is controlled by the parameter  $0 \leq \eta_u \leq 1$ . A low value would be adequate for functions where multimodality is not a concern, either because the positions of the maxima are already known by other means, because there can only be one, or because the different peaks are not too isolated (i.e. the contrast between the maximum and the saddle point towards the nearest detected peak is not large). A high value of  $\eta_u$  would help the algorithm locate the peaks, but it would render it slow. What constitutes a reasonable compromise between accuracy and computational resources depends, of course, on the details of the application being considered, and the adopted default value should not preclude the user from applying his/her own judgment.

## 2.2 Choice of $g(\mathbf{x})$

The prescription adopted for  $g(\mathbf{x})$  is arguably the heart of the method. The idea is to use the set of sample points computed so far to estimate the shape of the integrand. First, each point is assigned a hypercubical cell by the FiESTAS algorithm (see [8] and Appendix A). Let the volumes of these cells be denoted as  $v_i$ . Then,

$$g(\mathbf{x}) = \frac{\sum_{i=1}^N w_i \Pi_i(\mathbf{x})}{\sum_{i=1}^N w_i v_i} \quad (10)$$

with  $\Pi_i(\mathbf{x}) = 1$  if  $\mathbf{x}$  belongs to the  $i$ -th cell, and 0 otherwise.

The choice of weights  $w_i = 1$  corresponds to uniform sampling, and  $w_i = |f_i|$  corresponds to  $g(\mathbf{x}) \approx |f(\mathbf{x})|/I$ . This is the optimal probability distribution when the integrand shape is already well described by the sample points, i.e.  $f(\mathbf{x}) \approx \sum_{i=1}^N f_i \Pi_i(\mathbf{x})$ . In practice, convergence is faster if one uses instead

$$w_i^2 = \langle f^2 \rangle_i = \frac{1}{n_{\text{nei}}} \sum_{j=1}^{n_{\text{nei}}} f_j^2 \quad (11)$$

where the sum is performed over the  $n_{\text{nei}}$  neighbouring cells in the FIESTAS tessellation, including the cell  $i$  itself. This prescription tends to sample regions where  $|f|$  is large and/or varies on small scales (comparable to the sampling density), and therefore it combines in a simple way the advantages of importance and stratified sampling. Considering the values of  $f$  in adjacent cells is important in order to efficiently explore those regions where large gradients are found, such as newly discovered maxima or pronounced degeneracies.

### 2.3 Sampling from $g(\mathbf{x})$

Each step,  $n_F = (1 - \eta_u)n$  new points are generated from the distribution (10). The probability that each of these points belongs to cell  $i$  is given by

$$p_i = g_i v_i = \frac{w_i v_i}{\sum_{j=1}^N w_j v_j} \quad (12)$$

and therefore the cumulative distribution can be computed as

$$P_i = \sum_{j=1}^i p_j = \frac{P_{i-1} + w_i v_i}{P_N} \quad (13)$$

with  $P_0 = 0$ .

Sampling from  $P_i$  is then trivial. A uniform random number between 0 and 1 is generated, and the value of  $i$  is obtained from a binary search. The new point is created at a random location, uniformly distributed within cell  $i$ .

### 2.4 Evaluation of the result

From the  $n_F = (1 - \eta_u)n$  new points generated during the current step,

$$\hat{I}_s = \frac{1}{n_F} \sum_{i=1}^{n_F} \frac{f_i}{g_i} \quad (14)$$

and

$$\hat{\Delta}_s^2 = \frac{1}{n_F^2} \sum_{i=1}^{n_F} \left( \frac{f_i}{g_i} - \hat{I}_s \right)^2. \quad (15)$$

To estimate the value of  $I$ , the estimates of the last  $S$  steps are combined according to

$$\hat{I} = \frac{1}{S} \sum_{i=0}^{S-1} \hat{I}_{s-i} \quad (16)$$

with an associated error

$$\hat{\Delta}^2 = \frac{1}{S(S-1)} \sum_{i=0}^{S-1} \left( \hat{I}_{s-i} - \hat{I} \right)^2. \quad (17)$$

The number of steps  $S$  is set by the condition

$$\left| \hat{I}_{s-S} - \hat{I} \right| > 3 \min \{ \Delta_s, \Delta_S \}. \quad (18)$$

if  $S \geq 4$ . Else, the procedure is assumed not to have converged yet, and only the information from the last step is considered (i.e.  $\hat{I} = \hat{I}_s$  and  $\hat{\Delta} = \hat{\Delta}_s$ ).

## 2.5 Iteration

At the end of each step, the total number of new points  $n$  is increased by a fixed fraction  $\eta_n$ . Since the FIESTAS tree has to be updated once per step in order to estimate  $g(\mathbf{x})$ , a small value of  $\eta_n$  increases the overhead of the algorithm. However, a frequent update results in a more accurate sampling, and therefore a smaller number of function evaluations. A compromise should be sought for each particular problem, taking into account the computational cost of evaluating the integrand and setting  $\eta_n$  so as to minimize the total CPU time. Otherwise, the precise value of this parameter does not significantly affect the results.

## 2.6 Convergence

The main loop continues generating new points until the estimated relative error drops below the value of the tolerance parameter  $\epsilon$ ,

$$\hat{\Delta} < \epsilon |\hat{I}|. \quad (19)$$

Again, the default value should be interpreted as a guideline, subject to trial and error, common sense, and the specific requirements of the problem at hand. Note that, as mentioned above, the condition  $S \geq 4$  is also imposed

in order to ensure that the algorithm has actually converged. This criterion is only important in those few cases where a new maximum has just been discovered at the time  $\hat{\Delta}$  decreased below the requested tolerance.

### 3 Results

The performance of FIESTAS sampling has been tested by applying the algorithm to a series of toy multidimensional problems with known analytical solution. The integrands have been chosen to emphasize the issues of multimodality and the presence of non-linear degeneracies. I also consider the test suite proposed by Genz [9] and compare the results with those of other algorithms.

Two different aspects have been considered: the accuracy of the solution (the actual value of the integral  $\hat{I}$ , the quality of the error estimate  $\hat{\Delta}$ , and the ability to find all maxima) and the computational cost (in terms of the number of evaluations of the function  $f$  as well as the overhead in CPU time incurred by the FIESTAS tree).

All the tests have been carried out on a Pentium 4 processor with a clock rate of 3.2 GHz. The parameters of the algorithm are always set to their default values.

#### 3.1 Toy problem I

As a first example, the method is used to compute the integral of a linear combination of five multivariate Gaussians with different values of  $\sigma$ , located at arbitrary points in the  $xy$ -plane. The locations and dispersions are the same as in [5], i.e.  $x_i = \{-0.4, -0.35, -0.2, 0.1, 0.45\}$ ,  $y_i = \{-0.4, 0.2, 0.15, -0.15, 0.1\}$ , and  $\sigma_i = \{0.01, 0.01, 0.02, 0.03, 0.05\}$ , but each of the Gaussian components has been normalized in order to facilitate the counting of the number of peaks detected,

$$f(\mathbf{x}) = \sum_{i=1}^5 \frac{1}{(2\pi\sigma_i^2)^{D/2}} \exp \left[ -\frac{|\mathbf{x} - \mathbf{c}_i|^2}{2\sigma_i^2} \right] \quad (20)$$

where  $D$  is the dimensionality of the problem and  $\mathbf{c}_i$  is the centre of each Gaussian, given by  $x_i$  and  $y_i$ .

The correct value of the integral (performed from  $-1$  to  $1$  along all axes) is thus  $I \simeq 5$ . In the Bayesian framework, each of the maxima would correspond to a different solution. Since all the individual evidences are equal, these five solutions would be equally valid, and none of them would be preferred over



$\hat{I}$	$\hat{\Delta}$	$N$	$t_{\text{CPU}}$
3.982	0.037	3974	0.31
3.979	0.026	3603	0.27
5.095	0.047	3603	0.27
4.026	0.024	1990	0.14
3.958	0.032	3603	0.27
4.035	0.040	3603	0.27
4.032	0.039	2682	0.19
5.052	0.044	5325	0.42
4.144	0.039	3603	0.27
4.998	0.048	7127	0.57

Table 1

Results obtained by 10 independent runs of FIESTAS sampling for the toy problem I in two dimensions. First and second columns quote the estimates given by equations (16) and (17), respectively, followed by the total number of evaluations and the CPU time (in seconds) required by the algorithm.

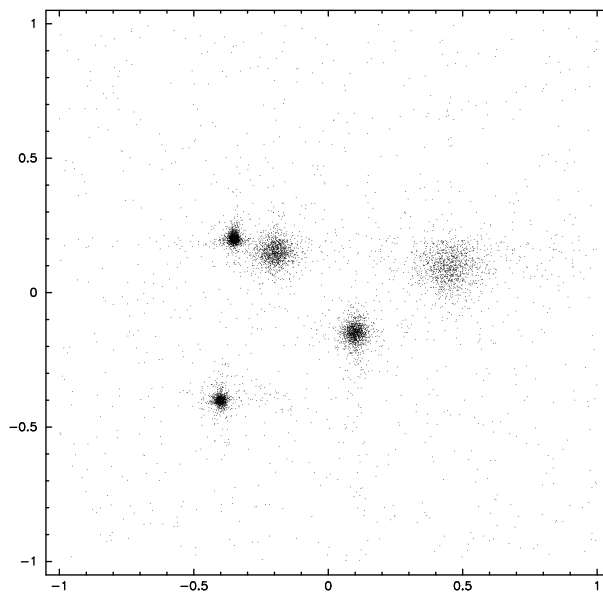


Fig. 1. Sample points used by the last run in Table 1

the others. From a frequentist point of view, the narrowest peaks should be favoured because they yield the highest values of the likelihood  $f$ .

The results of ten independent runs with different random seeds are listed in Table 1 for the two-dimensional case. In a larger set of one hundred runs, the number of peaks identified by the algorithm was three (4 times), four

$D$	$\langle \hat{I} \rangle \pm \sigma_I$	$\langle \hat{\Delta}/\Delta_{\text{true}} \rangle \pm \sigma_{\Delta}$	$\langle N \rangle \pm \sigma_N$	$\langle t_{\text{CPU}} \rangle \pm \sigma_t$
2	$4.490 \pm 0.485$	$0.767 \pm 1.836$	$4400 \pm 1397$	$0.34 \pm 0.12$
3	$3.910 \pm 0.309$	$2.128 \pm 3.254$	$18896 \pm 6003$	$2.87 \pm 1.00$
4	$4.161 \pm 0.325$	$0.722 \pm 3.254$	$86084 \pm 40075$	$24.64 \pm 12.07$
5	$3.525 \pm 1.362$	$1.441 \pm 2.796$	$213956 \pm 115194$	$104.41 \pm 59.64$
6	$1.790 \pm 1.577$	$0.796 \pm 2.186$	$482739 \pm 545036$	$380.49 \pm 487.6$
7	$2.001 \pm 1.615$	$1.343 \pm 2.383$	$2937754 \pm 3908586$	$4207.78 \pm 6332$
8	$1.501 \pm 1.094$	$0.421 \pm 4.539$	$6578407 \pm 5835573$	$13554.28 \pm 15750$

Table 2

Results for toy problem I in  $D$  dimensions, averaged over ten independent runs. Columns show  $D$ , the value of  $\hat{I}$ , the deviation of the estimated error  $\hat{\Delta}$  with respect to  $\Delta_{\text{true}}$  (see text), the number of evaluations, and the CPU time in seconds.

(56 times), and five (40 times). The narrow Gaussian at  $(-0.4, -0.4)$  was the most difficult to detect. Although it has the same dispersion as the second component at  $(-0.35, 0.2)$ , the latter is relatively close to another broader maximum, and thus it was much easier to identify. Neglecting the contribution from missed peaks, the estimated error always corresponded fairly well to the actual deviation with respect to the analytical solution. The number of sample points required to achieve convergence ranged from  $N = 1990$  to  $11557$ , and the run time was in all cases of the order of a few tenths of a second. The sample points used by the last run are plotted in Figure 1.

The dependence on the number of dimensions  $D$  is shown in Table 2. The five Gaussians are kept at the same locations in the  $xy$ -plane, with the same dispersions and normalized to unit evidence, but they refer now to  $D$  independent variables ranging from  $-1$  to  $1$ . Each entry on the table corresponds to the average and standard deviation of ten independent runs with different random seeds.

FiESTAS sampling typically discovers four or five peaks for  $D = 2$ , which leads to the average value  $\langle \hat{I} \rangle = 4.5 \pm 0.5$ . As the number of dimensions increases, detecting each of the isolated maxima becomes more difficult, and the value of  $\langle \hat{I} \rangle$  decreases. Note, however, that large dispersions  $\sigma_I$  indicate that the number of peaks detected may vary considerably between different runs. For instance, in  $D = 6$  dimensions, the algorithm stopped after finding only the broad maximum at  $(0.45, 0.1)$  in eight out of the ten runs, whereas all peaks were correctly identified in the other two.

The values of  $\langle \hat{\Delta}/\Delta_{\text{true}} \rangle$  correspond to the geometrical average of the estimated error  $\hat{\Delta}$  divided by the ‘true’ error  $\Delta_{\text{true}} = |\hat{I} - I'|$ , where  $I'$  is the

nearest integer to  $\hat{I}$ . More precisely,

$$\langle \hat{\Delta}/\Delta_{\text{true}} \rangle \equiv \exp \left[ \left\langle \ln \left( \frac{\hat{\Delta}}{|\hat{I} - I'|} \right) \right\rangle \right] \quad (21)$$

and

$$\sigma_{\Delta} \equiv \exp \left[ \left\langle \ln^2 \left( \frac{\hat{\Delta}}{|\hat{I} - I'|} \right) \right\rangle - \left\langle \ln \left( \frac{\hat{\Delta}}{|\hat{I} - I'|} \right) \right\rangle^2 \right]. \quad (22)$$

Since all of the Gaussians have unit evidence,  $I'$  corresponds to both the number of peaks detected and the analytical value of the integral after subtracting the contribution from the undetected components. The results obtained,  $\langle \hat{\Delta}/\Delta_{\text{true}} \rangle \sim 1 \pm 3$ , suggest that  $\hat{\Delta}$  is typically within a factor of three (above or below) from  $\Delta_{\text{true}}$  (see e.g. Table 1).

Finally, the exact number of evaluations required depends on the integrand being considered, the requested tolerance  $\epsilon$ , and, to a lesser extent, the values of the parameters  $\eta_u$  and  $\eta_n$ . What can be inferred from the data in Table 2 is that  $N$  grows exponentially with the number  $D$  of dimensions. Since the complexity of the FIESTAS tree is  $\mathcal{O}(N \log N)$  [8], the overhead in CPU time also grows exponentially.

### 3.2 Toy problem II

In order to assess the effect of non-linear degeneracies, let us also investigate the toy problem proposed by [10],

$$f(\mathbf{x}) = R(\mathbf{x}; \mathbf{c}_1, r_1, w_1) + R(\mathbf{x}; \mathbf{c}_2, r_2, w_2) \quad (23)$$

with

$$R(\mathbf{x}; \mathbf{c}, r, w) = \frac{1}{\kappa} \exp \left[ -\frac{(|\mathbf{x} - \mathbf{c}| - r)^2}{2w^2} \right]. \quad (24)$$

The function  $R(\mathbf{x})$  represents a ring of radius  $r$  and width  $w$ , centered at  $\mathbf{c}$ . The normalization  $\kappa$  is chosen so that  $\int R(\mathbf{x}) d\mathbf{x} = 1$ , and it can be expressed as

$$\kappa = \frac{D \pi^{D/2}}{\Gamma(D/2 + 1) \sqrt{2\pi w^2}} K_{D-1}(r, w) \quad (25)$$

where  $\Gamma$  denotes the Gamma function and  $K_D$  is given by the recursive formula

$$K_D(r, w) = r K_{D-1} + (D-1)w^2 K_{D-2} \quad (26)$$

with  $K_0 = 1$  and  $K_1 = r$ . The two rings are separated by 7 units along the  $x$ -axis, and their radii and widths are set to the values  $r_1 = 1$ ,  $r_2 = 2$ , and  $w_1 = w_2 = 0.1$ . The domain of integration ranges from  $-6$  to  $6$  in all dimensions.

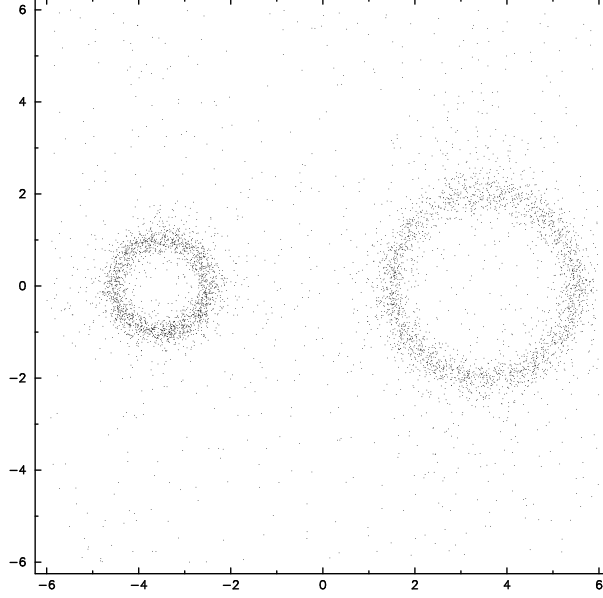


Fig. 2. Sample points for toy problem II in two dimensions.

$D$	$\langle \hat{I} \rangle \pm \sigma_I$	$\langle \hat{\Delta} / \Delta_{\text{true}} \rangle \pm \sigma_{\Delta}$	$\langle N \rangle \pm \sigma_N$	$\langle t_{\text{CPU}} \rangle \pm \sigma_t$
2	$1.990 \pm 0.023$	$1.792 \pm 2.991$	$6127 \pm 1012$	$0.47 \pm 0.09$
3	$2.001 \pm 0.017$	$1.411 \pm 2.137$	$29660 \pm 8082$	$4.60 \pm 1.35$
4	$1.984 \pm 0.027$	$0.938 \pm 2.700$	$72619 \pm 13452$	$20.03 \pm 4.04$
5	$1.897 \pm 0.304$	$1.199 \pm 2.473$	$179333 \pm 56814$	$83.86 \pm 29.15$
6	$1.491 \pm 0.501$	$0.933 \pm 3.181$	$327824 \pm 77328$	$249.65 \pm 64.26$
7	$1.196 \pm 0.397$	$0.782 \pm 2.524$	$735387 \pm 203382$	$552.04 \pm 169.9$
8	$1.381 \pm 0.480$	$0.675 \pm 2.206$	$4517358 \pm 3616613$	$11001.25 \pm 9788$

Table 3

Results for toy problem II.

The values of the integral computed by FIESTAS sampling are given in Table 3, and Figure 2 shows the sample points used by the algorithm in the two-dimensional case. The two rings have been correctly identified by the algorithm ten times out of ten up to  $D = 4$ . As the number of dimensions increases, the small ring occupies an exponentially smaller fraction of the integration domain with respect to the bigger structure, and the detection rate decreases. For  $D = 8$ , it is found in four of the ten runs.

The shape of the rings is always accurately recovered, which proves that the presence of strong degeneracies does not significantly affect the accuracy of the method. As in the previous examples, the estimated error is within a factor of three from the true value, and both the number  $N$  of evaluations and the overhead  $t_{\text{CPU}}$  grow exponentially with  $D$ .

Model	$D$	$N_{\text{Vegas}}$	$N_{\text{Suave}}$	$N_{\text{Divonne}}$	$N_{\text{Cuhre}}$	$N_{\text{FiEstAS}}$
I	2	52000	90000	18569	9165	3974
	3	45000	160000	193910	82677	14012
	4	175000	320010	285312	215271	168641
	5	314500	1270259	86567	1150695	204091
	6	-	120036	171526	3971451	224517
	7	-	280084	236217	26245785	11186912
	8	-	1840397	6619086	116242685	7640740
II	2	45000	60000	6321	19175	5325
	3	67500	90000	24700	213233	33236
	4	85000	230006	99300	8675253	58999
	5	137500	320015	189236	96301023	328797
	6	-	510046	219956	-	397883
	7	-	820128	364261	-	640912
	8	-	1120153	122915	-	1828970

Table 4

Number of evaluations required by the algorithms Vegas [6], Suave [7], Divonne [11], and Cuhre [12] for toy problems I and II in  $D$  dimensions. Empty entries indicate failure to converge within  $N = 10^9$  function calls. Results of a single run of FiEstAS sampling are quoted in the last column.

### 3.3 Comparison with other algorithms

The method presented here significantly outperforms the standard Monte Carlo with uniform probability, and its results are comparable to those of the bank sampling technique described in [10], at least for problems of low dimensionality. An important advantage of FiEstAS sampling is that prior information on the shape of  $f$  is not required, although it could be easily taken into account by simply adding the ‘bank’ points at any stage. On the other hand, comparison with the results of [5] suggests that clustered nested sampling may be more efficient for large  $D$ .

Tables 4 and 5 show the performance of the algorithms Vegas [6], Suave [7], Divonne [11], and Cuhre [12], as implemented in the Cuba library<sup>1</sup> [7], when applied to toy problems I and II. Table 4 quotes the number of integrand evaluations, and Table 5 shows the estimated value of the integral. FiEstAS

<sup>1</sup> <http://www.feynarts.de/cuba>

Model	$D$	$\hat{I}_{\text{Vegas}}$	$\hat{I}_{\text{Suave}}$	$\hat{I}_{\text{Divonne}}$	$\hat{I}_{\text{Cuhre}}$	$\hat{I}_{\text{FiEstAS}}$
I	2	5.01	5.00	4.95	5.00	4.06
	3	2.29	3.99	5.00	5.00	4.09
	4	1.03	3.99	3.64	2.00	4.56
	5	2.11	3.95	2.61	2.00	3.96
	6	-	0.99	2.01	2.00	1.00
	7	-	0.37	2.47	2.00	4.99
	8	-	1.80	0.58	1.99	3.01
II	2	2.00	2.00	2.02	2.00	2.00
	3	2.00	2.00	1.96	1.63	2.01
	4	1.95	2.00	2.01	1.54	1.98
	5	1.37	2.00	1.38	1.60	2.00
	6	-	1.26	1.20	-	2.01
	7	-	0.50	1.00	-	0.96
	8	-	0.30	0.51	-	0.99

Table 5

Estimates  $\hat{I}$  returned by the different algorithms (same runs as in Table 4).

sampling is quite competitive in terms of efficiency; the values of  $N$  are always comparable to or lower than those required by the other methods. Its main drawback, however, is the computational overhead; the CPU times in Tables 2 and 3 are much larger than those required by the other algorithms (a few minutes in the worst cases). In terms of accuracy, the number of detected maxima compares very favourably with the other methods. Moreover, the estimate  $\hat{I}$  is almost always close to the true value of the integral, ignoring the contribution from undetected peaks; among all the examples quoted, only the run for toy model I in four dimensions reported a wrong value.

In order to probe a wider variety of integrands, the different methods are also benchmarked with the Genz test suite [9], based on the following six function families:

(1) Oscillatory:

$$f_1(\mathbf{x}) = \cos(\mathbf{c} \cdot \mathbf{x} + 2\pi w_1)$$

(2) Product peak:

$$f_2(\mathbf{x}) = \prod_{d=1}^D \frac{1}{(x_d - w_d)^2 + c_d^{-2}}$$

(3) Corner peak:

$$f_3(\mathbf{x}) = (1 + \mathbf{c} \cdot \mathbf{x})^{-(D+1)}$$

(4) Gaussian:

$$f_4(\mathbf{x}) = \exp(-|\mathbf{c} \cdot (\mathbf{x} - \mathbf{w})|^2)$$

(5)  $C^0$ -continuous:

$$f_5(\mathbf{x}) = \exp(-\mathbf{c} \cdot |\mathbf{x} - \mathbf{w}|)$$

(6) Discontinuous:

$$f_6(\mathbf{x}) = \begin{cases} \exp(\mathbf{c} \cdot \mathbf{x}) & \text{if } x_1 < w_1 \text{ and } x_2 < w_2 \\ 0 & \text{otherwise} \end{cases}$$

The parameter vector  $\mathbf{c}$  sets the overall difficulty of the problem. Its components are chosen as uniform random numbers from 0 to 1, and then they are normalized according to

$$\|\mathbf{c}\|_1 \equiv \sum_{d=1}^D c_d = 50 \quad (27)$$

except for the oscillatory integrand, for which I set  $\|\mathbf{c}\|_1 = 5$ . The components of  $\mathbf{w}$  are uniform random numbers between 0 and 1. In general terms, they do not affect the difficulty of the problem, since they only control the location of the features (e.g. maxima) of  $f$ . They play an important role, though, for the oscillatory family, because  $w_1$  sets the precise value of the integral, which can be arbitrarily close to zero. This poses a problem for FiESTAS sampling (and, more generally, to pure Monte Carlo algorithms) because a large number of evaluations is required to achieve the cancellation of positive and negative terms with the desired accuracy.

Results of the Genz test in 2, 5, and 8 dimensions are quoted in Table 6 for the different algorithms. Each entry shows the average number of evaluations over 20 random realizations of each family. In order to speed up the test, we impose a maximum number of evaluations  $N_{\max} = 10^6$ . It is clear from the comparison that every algorithm has its own strengths and weaknesses. Some methods are better suited to solve a particular class of problems, and some others perform better in many dimensions. In general terms, the results presented here suggest that FiESTAS sampling could be a good choice for complicated integrands of low to moderate dimensionality.

## 4 Conclusions

This paper presents a new algorithm to carry out numerical integration in multiple dimensions or, in a Bayesian context, sample from the posterior distribution and compute the evidence for the assumed model. The method, dubbed ‘FiESTAS sampling’, is a variant of importance sampling where the

Genz	$D$	$\langle N_{\text{Vegas}} \rangle$	$\langle N_{\text{Suave}} \rangle$	$\langle N_{\text{Divonne}} \rangle$	$\langle N_{\text{Cuhre}} \rangle$	$\langle N_{\text{FiEstAS}} \rangle$
1	2	<sup>(2)</sup> 97056	66500	2954	195	<sup>(3)</sup> 165913
	5	18350	42000	2917	819	<sup>(1)</sup> 22110
	8	68500	51500	3697	3315	<sup>(1)</sup> 57411
2	2	5625	20500	3186	2190	1096
	5	9100	31500	10624	477422	16559
	8	12300	40000	23326	<sup>(19)</sup> 469625	57301
3	2	4500	13500	1690	195	395
	5	4500	20000	3970	819	3231
	8	7000	20000	5495	3315	11505
4	2	5750	23000	7137	1768	2037
	5	16950	42000	30582	57630	32559
	8	30200	64000	<sup>(3)</sup> 52943	<sup>(10)</sup> 549185	183580
5	2	5375	22500	4542	4407	1595
	5	10025	33500	19747	<sup>(17)</sup> 671853	21697
	8	16650	42500	<sup>(1)</sup> 40161	<sup>(20)</sup> -	92247
6	2	7825	32500	6994	50453	3056
	5	18025	87008	<sup>(1)</sup> 64257	84766	25940
	8	<sup>(1)</sup> 36079	103006	<sup>(4)</sup> 152156	<sup>(12)</sup> 604435	115577

Table 6

Average number of evaluations required by each algorithm for the test suite proposed by Genz [9]. The small numbers in parentheses indicate the number of runs where the desired accuracy was not reached for  $N = 10^6$ .

weight of each point is computed with the help of the Field Estimator for Arbitrary Spaces [8].

Its performance has been tested for several toy problems with known analytical solution, specifically designed to contain multimodal distributions and significant degeneracies. The results suggest that FiEstAS sampling provides an interesting alternative to other methods for problems of low dimensionality. In particular, it is able to discover most isolated maxima of the integrand, and it can perfectly sample from distributions with pronounced degeneracies. As the number of dimensions increases, the ability of the algorithm to identify peaks decreases and the required computational resources (in terms of both time and memory) grow exponentially.



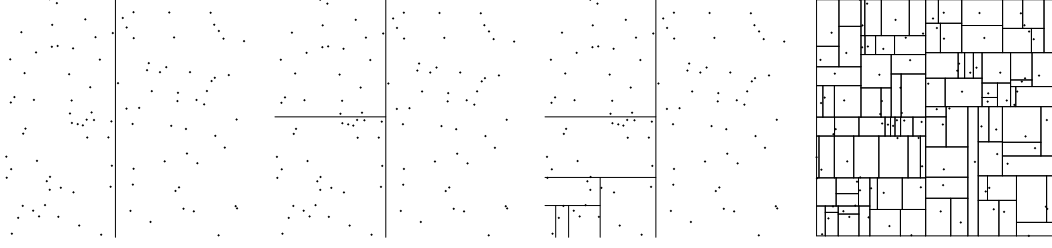


Fig. A.1. Fiestas algorithm applied to a random uniform distribution of 100 points in two dimensions.

## A Fiestas

The Field Estimator for Arbitrary Spaces (Fiestas) is a technique to estimate the continuous (probability) density field underlying a given distribution of data points. Particular attention is paid to avoid imposing a metric on the data space. Indeed, the problem may actually be regarded as *computing* the appropriate metric, given the data.

Fiestas assigns each point a volume  $v_i$  by means of a  $k-d$  tree. The space is recursively divided, one dimension at a time, until there is only one single data point in each node. The original implementation, described in [8], is heavily oriented towards a particular problem, namely the estimation of densities in phase space (a non-Euclidean, six-dimensional space composed of three-dimensional positions and velocities). Here I use a more general version of the algorithm, where the dimension less likely to arise from a uniform distribution is selected for splitting at each step (very similar to the Shannon entropy approach followed by [13]). More precisely, a histogram with  $B = 1 + \sqrt{N_{\text{node}}}$  bins is built for each dimension, and the log-likelihood

$$L_d = \ln(N_{\text{node}}!) - N_{\text{node}} \ln(B) - \sum_{b=1}^B \ln(n_{bd}!) \quad (\text{A.1})$$

is computed, where the indices  $1 \leq d \leq D$  and  $1 \leq b \leq B$  denote the dimension and the bin number, respectively,  $n_{bd}$  is the number of points in each bin, and  $N_{\text{node}}$  is the total number of points in the node. In order to encourage a similar number of divisions along all dimensions, I add to  $L_d$  the number of times  $s_d$  that the  $d$ -th axis has already been split,

$$L'_d = L_d + s_d. \quad (\text{A.2})$$

The dimension with smaller  $L'$  is divided at the point  $x_{\text{split}} = (x_l + x_r)/2$ , where  $x_l$  is the maximum  $x$  of all points lying on the 'left' side ( $b \leq b_{\text{split}}$ ) and  $x_r$  is the minimum  $x$  of the points lying on the 'right' ( $b > b_{\text{split}}$ ) side. The bin  $1 \leq b_{\text{split}} < B$  is chosen in order that the number of points on each side is as close as possible to  $N_{\text{node}}/2$ .

The procedure is illustrated in Figure A.1, where successive steps of the algorithm are plotted for a random realization of 100 uniformly-distributed data points in two dimensions. For this particular realization, the dimension with smaller  $L'$  turned out to be the  $x$ -axis (first panel in the figure). Then, for the points on the left side, the  $y$ -axis was selected for splitting (second panel). The same axis was chosen again for the bottom region, and the sequence continued by dividing further along the  $x$ -,  $y$ -,  $x$ -, and  $x$ -axes. At this moment (third panel in the figure) there is only one point on the left side, and control returns to the parent node in order to split the right branch (containing, in this case, two points). The final tessellation obtained by this method is shown on the last panel.

## References

- [1] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., Numerical recipes in FORTRAN. The art of scientific computing, Cambridge University Press, 1992.
- [2] Mackay, D. J. C., Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003.
- [3] Skilling, J., Nested Sampling, in *AIP Conference Series*, edited by Fischer, R., Preuss, R., and Toussaint, U. V., volume 735, 2004.
- [4] Shaw, J. R., Bridges, M., and Hobson, M. P., MNRAS **378** (2007) 1365.
- [5] Feroz, F. and Hobson, M. P., astro-ph/0704.3704 (2007).
- [6] Lepage, G. P., Journal of Computational Physics **27** (1978) 192.
- [7] Hahn, T., Computer Physics Communications **168** (2005) 78.
- [8] Ascasibar, Y. and Binney, J., MNRAS **356** (2005) 872.
- [9] Genz, A., A package for testing multiple integration subroutines, in *Numerical integration*, edited by Keast, P. and Fairweather, G., 1986.
- [10] Allanach, B. C. and Lester, C. G., hep-ph/0705.0486 (2007).
- [11] Friedman, J. H. and Wright, M. H., ACM Trans. Math. Softw. **7** (1981) 76.
- [12] Berntsen, J., Espelid, T. O., and Genz, A., ACM Trans. Math. Softw. **17** (1991) 437.
- [13] Sharma, S. and Steinmetz, M., MNRAS **373** (2006) 1293.